

腾讯连连开发说明

一、测试用三元组申请

打开腾讯云平台 <https://console.cloud.tencent.com/iotexplorer>，登录账号。 打开公共实例



新建一个项目后进入项目页面新建一个产品，产品品类自行设置。

新建产品

产品名称 *

支持中文、英文、数字、下划线、空格（非首尾字符）、中英文括号、-、@、\ /的组合，最多不超过40个字符

产品品类

设备类型 设备 网关 子设备

认证方式 密钥认证 证书认证

通信方式 Wi-Fi 2G/3G/4G 5G BLE LoRaWAN 其它 ⓘ

数据协议 数据模板 自定义遗传

描述

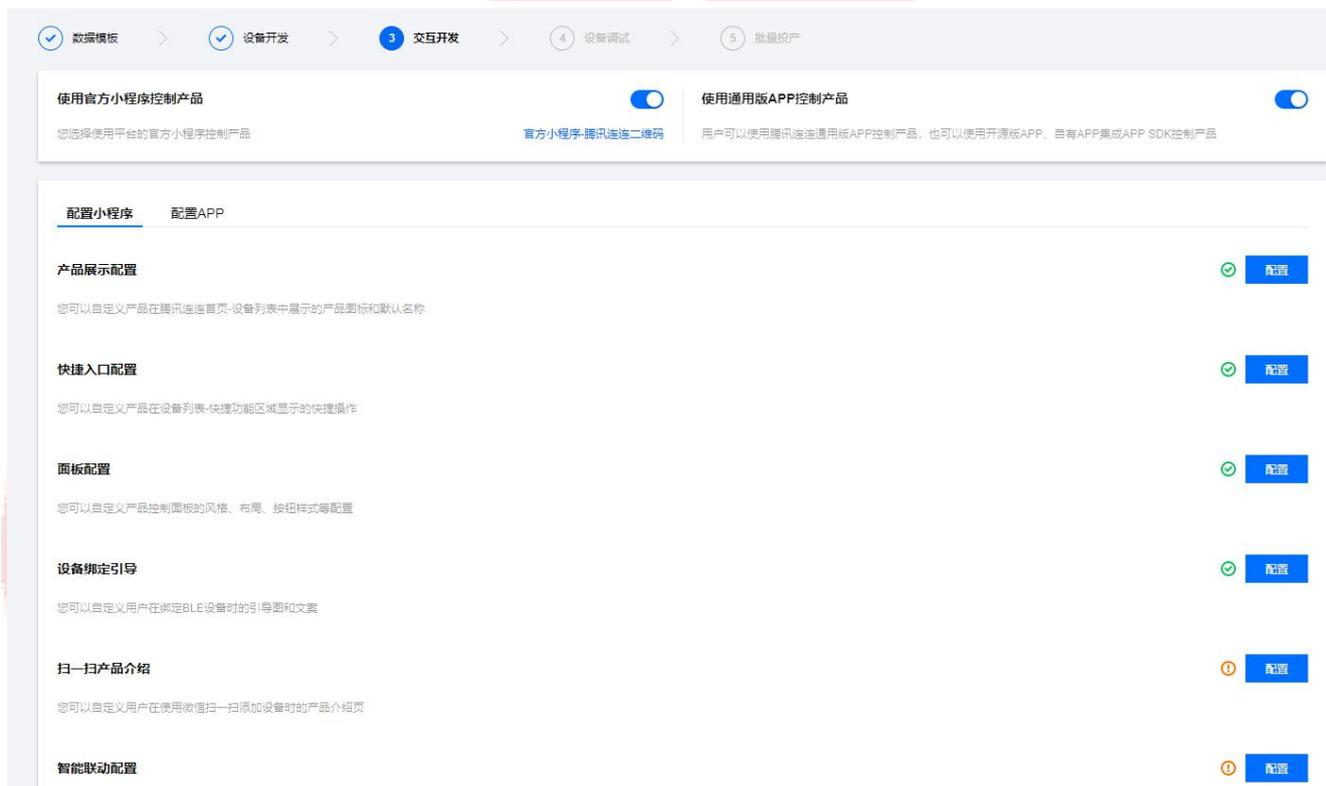
最多不超过80个字符

进入产品界面，设备开发界面选择基于标准蓝牙协议开发

产品开发 / 测试灯



交互开发根据需要自行设置



进入设备调试界面新建一个设备，点击设备名称进入设备信息界面



该界面的设备名称，设备密钥和产品 ID 即为该设备的三元组。

二、腾讯连连程序开发说明

1. 应用实例生成

进入产品的数据模版界面，根据需要增删功能后点击查看 JSON，将 JSON 文件下载下来。





根据 json 转换文档说明(script \ json \ readme.md)用 interpret_dt_ble.py 文件将 json 文件转换为应用代码 ble_qiot_template.c 和 ble_qiot_template.h。

2. 应用开发

将这两个文件替换掉 SDK 的 app/common/third_party_profile/Tecent_LL/include 下的 ble_qiot_template.h 和 app/common/third_party_profile/Tecent_LL/lib 下的 ble_qiot_template.c 文件。

ble_qiot_template.c 是根据控制台所添加的功能转换生成的 C 模版文件，例如本文档实例生成的 JSON 转换的 C 模版文件代码如下：

```
#include "ble_qiot_property.h"

static int ble_property_power_switch_set(const char *data, uint16_t len)
{
    return 0;
}

static int ble_property_power_switch_get(char *data, uint16_t buf_len)
{
    return sizeof(uint8_t);
}

static int ble_property_color_set(const char *data, uint16_t len)
{
    return 0;
}

static int ble_property_color_get(char *data, uint16_t buf_len)
{
    return sizeof(uint16_t);
}

static ble_property_t sg_ble_property_array[BLE_QIOT_PROPERTY_ID_BUTT] = {
    {ble_property_power_switch_set, ble_property_power_switch_get, BLE_QIOT_PROPERTY_AUTH_RW, BLE_QIOT_DATA_TYPE_BOOL},
    {ble_property_color_set, ble_property_color_get, BLE_QIOT_PROPERTY_AUTH_RW, BLE_QIOT_DATA_TYPE_ENUM},
};
```

该模版所对应的是灯的开关功能和颜色功能。你只需在 ble_property_power_switch_set 函数中实现灯开关控制，在 ble_property_power_switch_get 函数中实现灯状态获取，ble_property_color_set 函数中实现灯的颜色设置，在 ble_property_color_get 函数中实现灯的颜色获取即可。

功能实现实例：

```
static bool sw_state = 0;
static uint16_t led_color = 0;
static int ble_property_power_switch_set(const char *data, uint16_t len)
{
    sw_state = data[0];
    if (sw_state){
        open_light();
    }else{
        close_light();
    }
    return 0;
}

static int ble_property_power_switch_get(char *data, uint16_t buf_len)
{
    data[0] = sw_state;
    return sizeof(uint8_t);
}

static int ble_property_color_set(const char *data, uint16_t len)
{
    uint16_t test_enum = 0;

    memcpy(&test_enum, data, sizeof(uint16_t));
    led_color = ntohs(test_enum);
    if (led_color == 0) {
        set_led_red();
    } else if (led_color == 1) {
        set_led_green();
    } else if (led_color == 2) {
        set_led_blue();
    } else {
        printf("unknow color enum");
    }
    return 0;
}

static int ble_property_color_get(char *data, uint16_t buf_len)
{
    uint16_t test_enum = 0;

    printf("get led color:%d", led_color);
    test_enum = htons(led_color);
    memcpy(data, &test_enum, sizeof(uint16_t));

    return sizeof(uint16_t);
}

static ble_property_t sg_ble_property_array[BLE_QIOT_PROPERTY_ID_BUTT] = {
    {ble_property_power_switch_set, ble_property_power_switch_get, BLE_QIOT_PROPERTY_AUTH_RW, BLE_QIOT_DATA_TYPE_BOOL},
    {ble_property_color_set, ble_property_color_get, BLE_QIOT_PROPERTY_AUTH_RW, BLE_QIOT_DATA_TYPE_ENUM},
};
```

更多功能实现实例请参考应用实例：template 实例。

3. SDK 配置

打开 earphone/include/app_config.h 文件，将#define CONFIG_APP_BT_ENABLE 这个宏打开，同时将 LL_SYNC_EN 配置为 1。

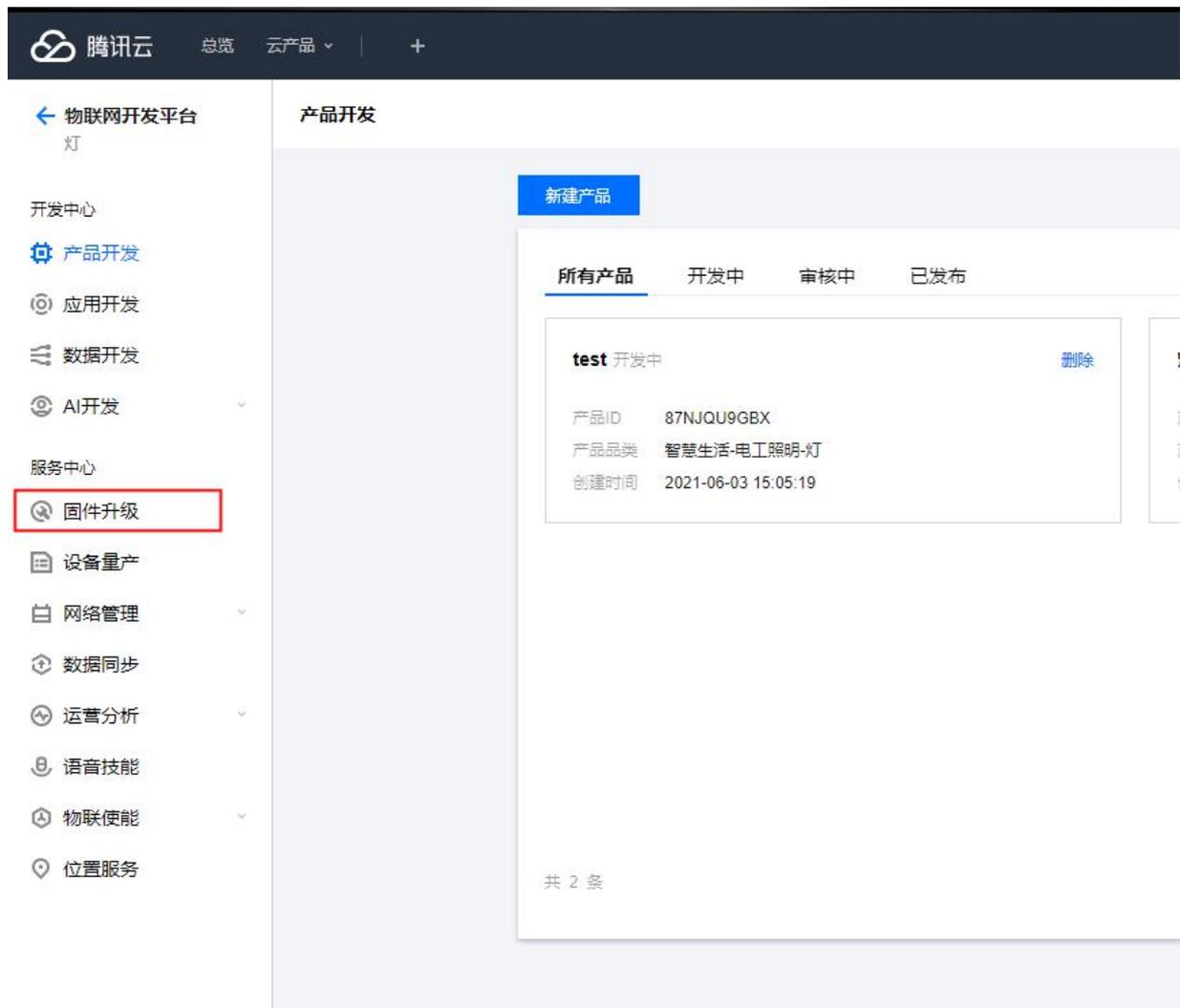
```
16
17
18 //*****//
19 //                               AI配置                               //
20 //*****//
21 #define CONFIG_APP_BT_ENABLE
22
23
24 #ifndef CONFIG_APP_BT_ENABLE
25 #define TME_EN 0
26 #define TRANS_DATA_EN 0
27 #define RCSP_BTmate_EN 0
28 #define RCSP_ADV_EN 0
29 #define XM_MMA_EN 0
30 #define LL_SYNC_EN 1
31 #else
32 #define TME_EN 0
33 #define TRANS_DATA_EN 0
34 #define RCSP_BTmate_EN 0
35 #define RCSP_ADV_EN 0
36 #define XM_MMA_EN 0
37 #define LL_SYNC_EN 0
38 #endif
39
40
41 #include "board_config.h"
42
43 #if CONFIG_APP_OTA_ENABLE
44 #if RCSP_ADV_EN || RCSP_BTmate_EN
45 #define RCSP_UPDATE_EN 1 //是否支持rcsp升级
46 #else
47 #define RCSP_UPDATE_EN 0 //是否支持rcsp升级
48 #endif
49 #endif
50
51 #if CONFIG_UPDATE_WITH_MD5_CHECK_EN
52 #define UPDATE_MD5_ENABLE 1
53 #else
54 #define UPDATE_MD5_ENABLE 0
55 #endif
56
57 NORMAL master ~/master/SDK/apps/earphone/include/app_config.h
58 "~/master/SDK/apps/earphone/include/app_config.h" 465L, 16092C written
```

4. OTA 升级文件上传

上传 OTA 升级文件时需要先将 ble_qiot_config.h 里这个宏的版本号进行更改，此处的版本需要高于原有固件的版本才能进行 OTA 升级。修改好后需要完成一次编译。

```
ble_qiot_config.h
67 #define BLE_QIOT_LOG_PRINT(...) printf(__VA_ARGS__)
68
69 #define BLE_QIOT_LLSYNC_STANDARD 1 // support llsync standard
70 #if BLE_QIOT_LLSYNC_STANDARD
71 // some users hope to confirm on the device before the binding, set BLE_QIOT_SECURE_BIND is 1 to enable the secure
72 // binding and enable secure bind in iot-explorer console. When the server is bound, the device callback ble_secure_bind
73 // will be triggered, the user agree or refuse connect by ble_secure_bind_user_confirm(). If the device does not respond
74 // and the connection timeout, or the user cancel the connection in Tencent Lianlian, a notify will received in function
75 // ble_secure_bind_user_notify().
76 #define BLE_QIOT_SECURE_BIND 0
77 #if BLE_QIOT_SECURE_BIND
78 #define BLE_QIOT_BIND_WAIT_TIME 60
79 #endif //BLE_QIOT_SECURE_BIND
80
81 // some sdk info needs to stored on the device and the address is up to you
82 #define BLE_QIOT_RECORD_FLASH_ADDR 5
83
84 // define user develop version, pick from "a-zA-Z0-9-._" and length limits 1~32 bytes.
85 // must be consistent with the firmware version that user write in the iot-explorer console
86 // refer https://cloud.tencent.com/document/product/1081/40296
87 #define BLE_QIOT_USER_DEVELOPER_VERSION "0.0.1"
88
89 #define BLE_QIOT_SUPPORT_OTA 1 // 1 is support ota, others not
90 #if BLE_QIOT_SUPPORT_OTA
91 #define BLE_QIOT_SUPPORT_RESUMING 0 // 1 is support resumming, others not
92 #if BLE_QIOT_SUPPORT_RESUMING
93 // storage ota info in the flash if support resumming ota file
94 #define BLE_QIOT_OTA_INFO_FLASH_ADDR 10
95 #endif //BLE_QIOT_SUPPORT_RESUMING
96
97 #define BLE_QIOT_TOTAL_PACKAGES 0x12 // the total package numbers in a loop
98 #define BLE_QIOT_PACKAGE_LENGTH 0xb0 // the user data length in package, ble_get_user_data_mtu_size() - 3 is the max
99 #define BLE_QIOT_RETRY_TIMEOUT 0x20 // the max interval between two packages, unit: second
100 // the time spent for device reboot, the server waiting the device version reported after upgrade, unit: second
101 #define BLE_QIOT_REBOOT_TIME 5
102 #define BLE_QIOT_PACKAGE_INTERVAL 0xa // the interval between two packages send by the server
103 // the package from the server will storage in the buffer, write the buffer to the flash at one time when the buffer
104 // overflow. reduce the flash write can speed up file download, we suggest the BLE_QIOT_OTA_BUF_SIZE is multiples
105 // of BLE_QIOT_PACKAGE_LENGTH and equal flash page size
106 #define BLE_QIOT_OTA_BUF_SIZE (512 * 4)
107 #endif //BLE_QIOT_SUPPORT_OTA
108 #endif //BLE_QIOT_LLSYNC_STANDARD
109
110 #define BLE_QIOT_LLSYNC_CONFIG_NET (!BLE_QIOT_LLSYNC_STANDARD) // support llsync configure network
111
112 #if (1 == BLE_QIOT_LLSYNC_STANDARD) && (1 == BLE_QIOT_LLSYNC_CONFIG_NET)
113 #error "llsync standard and llsync configure network is incompatible"
114 #endif
115
116 #ifdef __cplusplus
117 }
118 #endif
119
120 #endif // QCLOUD_BLE_QIOT_CONFIG_H
NORMAL master ~/master/SDK/apps/common/third_party_profile/Tecent_LL/include/ble_qiot_config.h
```

进入腾讯物联网平台，点击固件升级，进入该页面后点击添加固件



填写好固件名称，在所属产品那选择该固件所对应的产品，填写好固件版本号，该版本号需要与上述程序中的 ble_qiot_config.h 里的 BLE_QIOT_USER_DEVELOPER_VERSION 版本号对应。

添加新固件

固件名称 *

支持中文、英文大小写、数字、部分常用符号（下划线，减号，括弧），必须以中文、英文或数字开头，长度不超过32个字符

所属产品 *

固件版本号 *

仅支持英文字母、数字、点、中划线和下划线，长度限制1~32

选择固件 *

仅支持 bin, tar, gz, zip 类型的文件,文件大小不能超过1024MB

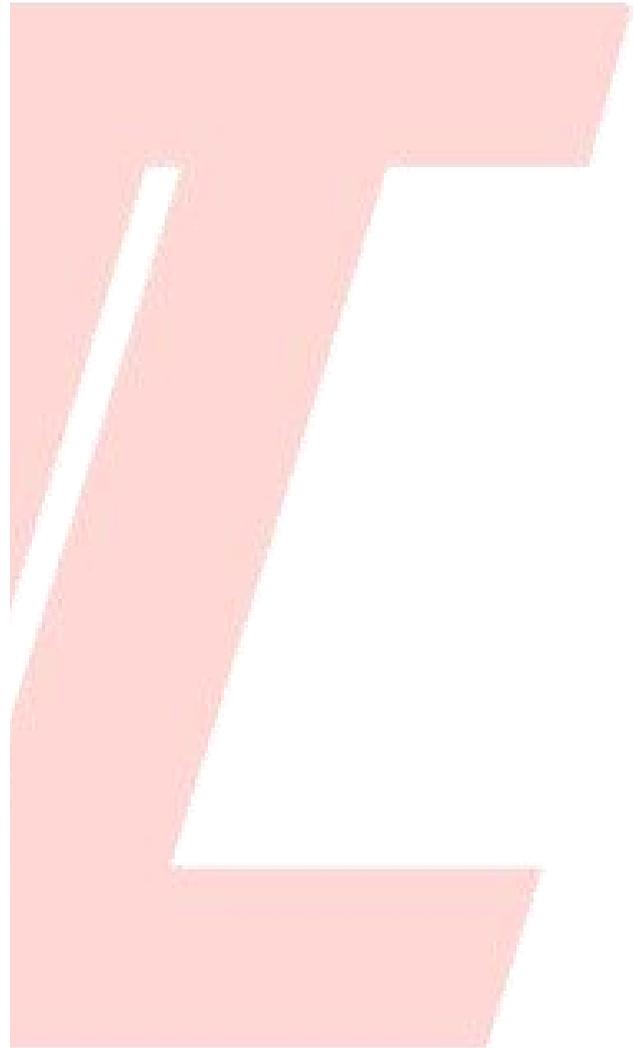
固件描述

对本次上传的固件进行描述和记录, 请输入0-100个字符

点击选择固件,OTA 固件文件为 db_update_data.bin, 该文件的路径在:CPU\BR30\tools\db_update_data.bin, 上传成功后点击保存即完成了 OTA 文件升级。

5. 手机端基本操作

使用微信搜索“腾讯连连”找到小程序。在小程序界面点击添加设备。



在添加设备界面，如果设备正常运行，可以发现设备的名字。点击连接。



连接成功后会显示设备略缩图，点击略缩图进入设备的控制面板。



在此处可以进行设备的控制操作。点击此处可以进行设备的设置。



点击固件升级可以检查是否有新版本固件，若存在则可以进行 OTA 升级。

